Named Entity Recognition by Neural Prediction

Nouha Hammadi, Abdel Belaïd and Yolande Belaïd
Université de Lorraine-CNRS-LORIA, Vandoeuvre-Lès-Nancy, France
hammadi.nouhaa@gmail.com, {abdel,yolande}.belaid@loria.fr

Abstract— *Named entity recognition (NER) remains a very challenging problem essentially when the document, where we perform it, is handwritten and ancient. Traditional methods using regular expressions or those based on syntactic rules, work but are not generic because they require, for each dataset, additional work of adaptation. We propose here a recognition method by context exploitation and tag prediction. We use a pipeline model composed of two consecutive BLSTMs (Bidirectional Long-Short Term Memory). The first one is a BLSTM-CTC coupling to recognize the words in a text line using a sliding window and HOG features. The second BLSTM serves as a language model. It cleverly exploits the gates of the BLSTM memory cell by deploying some syntactic rules in order to store the content around the proper nouns. This operation allows it to predict the tag of the next word, depending on its context, which is followed gradually until the discovery of the named entity (NE). All the words of the context are used to help the prediction. We have tested this system on a private dataset of Philharmonie de Paris, for the extraction of proper nouns within sale music transactions as well as on the public IAM dataset. The results are satisfactory, compared to what exists in the literature.*

Keywords: BLSTM; CTC; Historical documents; Named entity; Language model; Tag prediction

1. INTRODUCTION

Named entity recognition is a topic largely addressed in automatic natural language processing (NLP) which seeks to locate and extract names in text related to persons, organizations, addresses, expressions of times, monetary values, percentages, etc. It is experiencing a relatively large growth in information retrieval and document indexing on the Internet. It can be found frequently in printed administrative documents where, after having passed the document to the OCR, one will look in invoices, for example, names of customers, suppliers, billing dates or amounts of products purchased [7]. Its use in handwritten historical documents is very crucial to read the text and extract useful information. The last competition organized at ICDAR 2017 [10] shows the interest of the subject.

Recognizing known named-entities (NEs) is relatively simple and accurate, while recognizing novel NEs requires recognizing context and/or word-internal features. Named entities are often not simply singular words, but are chunks of text. Therefore, some chunking or parsing prediction model is required to predict whether a group of tokens belong to the same entity [19]. Some algorithms like Left to right decoding, Viterbi and beam search have been employed as chunking algorithms [18].

Proper names are more complicated to recognize than NE, especially when there is no lexicon of these names. The left and right contexts of these names are the only real guides to locate and identify them, such as introductory formulas like “Mr”, “Mrs.”, “Bought by” or affiliate forms on the left side, such as “son of”, “student of”, “resident of”, “trader to”, etc.

In this project, we seek to develop a generic method for proper name extraction in scanned documents, that is not guided by a prior knowledge or fixed contexts, but rather open to any new form of presentation of the entity. The study is based on a request from the Museum of Music of the Philharmonie de Paris which wishes to mine its ancient funds of sale transactions of instruments in order to discover prestigious personalities and instrument names appearing in these transactions over time. The collections represent more than 7,000 art instruments and works. The documents proposed for study come from the workshop of Parisian Lutheran Gand, Bernardel, Caressa et Français. This fund covers a century and a half of history, from 1816 to 1944. This violin workshop, founded by Nicolas Lupot in Paris in 1796, knew an exceptional destiny due to the figure of its founder and those of its successors, the importance and prestige of its clientele, excellence in instrument manufacture, expertise, restoration and trade in ancient instruments.

The fund transactions shows the sales, therefore the value attributed to the instruments, as well as the references to restorations and transformations made to these instruments. Furthermore, the search is not easy in these archives because the funds represents 11,000 views with documents that have different structures, the text is handwritten and typed by different hands, and the instruments are not listed in index. The search by text recognition would therefore make a "transverse" search.

Each view corresponds to a double page of an open register. Each page contains a series of transactions on one or many lines (see Figure 1).

Our work focuses on NER by combining part of speech tagging with predefined word categories, appearing in the sales transaction lines (see Figure 2). The words of the

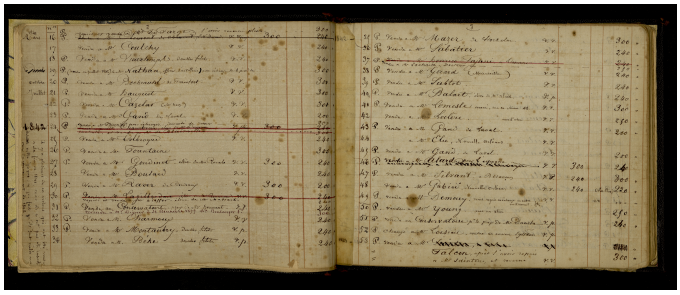


Fig. 1: Music sales database image example.

context to be recognized are those surrounding the proper nouns "Blachère" and "Maurice Thilhades".

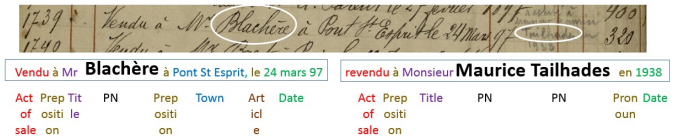


Fig. 2: Music sale transaction line example. The top line represents the image of the transaction. The middle line gives the transcript and the bottom line gives the tags of the words surrounding left and right, the two proper nouns searched for.

Our contribution to extract The NEs is summarized in the following points: 1) to label the named entities, 2) to label the remaining words that can help characterize the structure of the sentence either by using a tagger (Brill [4], for example)), or by manual labelling.

The remainder of the paper is organized as follows: Section II presents the related methods and their differences with our method. Section III describes the proposed method in details. The experiments performed with their results are described in section IV. Finally, a conclusion in section V provides a synthesis of the work and gives future trends.

2. Related work

As shown by [30], deep learning took an important step in the processing of sequences in pattern recognition. The recurrent models are particularly adapted to the problem which we are facing. They are successfully applied to the recognition of writing [21], [8]. They possess several advantages such as memorizing a specific context but the problem with these networks is that they can look back only a few steps regardless of the learning method employed due to the problem of vanishing gradient [3]. We look at a rapid reduction of error during back-propagation; The weights learned will take into account the recent context and not the distant context. In our case, we need to remember several previous steps so that we can have the maximum context (or the farthest and informative context) to decide. As a solution to this problem, Hochreiter et al. [17] propose the addition to the RNN, LSTM memory blocks which can retain information indefinitely. Lample et al. [20] proposed a neural architecture based on BLSTMs and conditional random fields (CRF). The model relies on character-based word representations and unsupervised word representations learned from non-annotated corpora. Graves et al. [16] presented a BLSTM to improve the results over LSTM networks. A BLSTM is composed of two RNNs with LSTM units. For each time step, a decision is made combining the output of two networks, taking advantage of the left and right context. The outputs of these two networks are then combined via a Softmax decision layer which provides later character probabilities in addition to a non-decisional class. This decision step is called Connectionist Time Classification (CTC) which allows the labelling of non-segmented data [12]. CTC is an objective function designed for labelling sequences with RNNs. It causes the network to directly map input sequences to an estimate of the conditional probabilities of possible labelling. It does not require the data to be pre-segmented. Finally, an alignment of the obtained sequence can be carried out by using a conventional dynamic programming algorithm or by developing a decoding via the integration of a sentence model in the form of a language statistical model (n-gram). Bideault et al. [6] present a word tracking system based on a BLSTM/HMM hybrid architecture. It relies on the strong discriminatory decisions of the BLSTM-CTC and using an HMM as a sequence model constrained by high-level information such as lexicons and/or language models. The decoding of a text line is conventionally achieved using Viterbi algorithm. This system uses the posterior probabilities calculated by BLSTM-CTC as a score to accept/reject the hypothesis. Stuner el al [28] propose a method for coupling a decoding without lexicon to a lexicon verification method. The idea is to accept a word if it belongs to the lexicon, otherwise reject it. This method is a combination method of BLSTM networks. It works at the level of a word image and the rejections of a level (BLSTM) are treated by the next level. The complementary aspect of the networks is made by a random initialization of the networks.

3. Proposed approach

The system proposed is composed of two BLSTMs (see. Figure 3). The first BLSTM, inspired by [13], is used for the recognition of the handwritten line. It is of type BLSTM-CTC because the context of both sides of a character is useful to improve the recognition. The second BLSTM is a language model that plays both the role of a word tagger and a context recognizer from the words (i.e. tags).

3.1 Text line recognition

We were inspired by Graves et al. system [13], where a sequence of N characteristic vectors is extracted by a sliding window approach with a width of one pixel and nine geometric features at each window position. The sequence

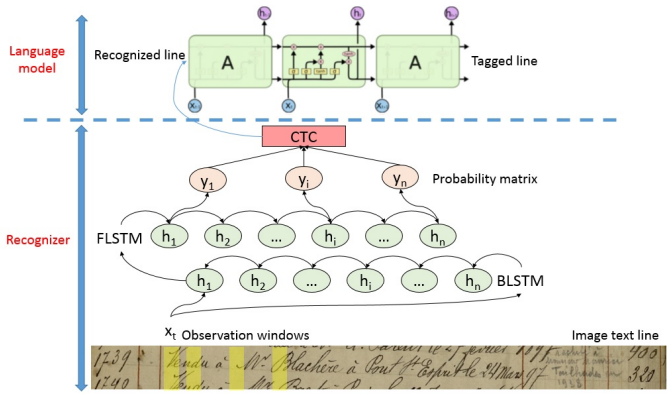


Fig. 3: Operating system.

of characteristic vectors is swept back and forth by the hidden layers of the BLSTM. For each time step t, a characteristic vector which presents the input of the network will be processed by the BLSTM layers. The BLSTM layers produce a probability distribution on the character transcripts by combining the results of the two layers (LSTM forward and backward) and taking advantage of the left and right context.

Our system is based on the same principle but the setting is different:

- At the level of feature extraction: instead of a nine-dimensional vector, we have used a HOG (Histogram of Oriented Gradients) descriptor which performs the gradient of an image and then traverses the image to fill a histogram of orientations.
- For decoding: find the most likely path without a dictionary combination and a language model.

The architecture consists of a single hidden layer containing 64 hidden states. The output layer contains 80 neurons: 26 for lowercase characters, 26 for uppercase characters and 28 for special characters. For each time step, the characteristics vector has 32 components. We trained the network with a gradient descent algorithm and a learning rate equal to $1e^{-2}$. The decoding is done by a beam search algorithm.

3.2 Tagging approach

After recognition of the entire line, a tagger is used to label each word. This tagger makes it possible to transform a sequence of words into another sequence of classes which helps to locate the searched word.

- Example: Entrance sequence: “vendu à Mr Martin le 12 juin 1864” (“sold to Mr Martin on June 12, 1864”) with the following classes:
  - `sold: Sale (VD)`
  - `to: Preposition (PRP)`
  - `Mr: Politeness (FP)`
  - `Martin: Proper Noun (NP)`
  - `the: Article (AR)`
  - `12: Day (DD)`
  - `June: Month (MM)`
  - `1864: Year (AA)`
- Output sequence: “vendu”/VD “à”/PRP “Mr”/FP “Martin”/NP “le”/AR “12”/JJ “juin”/MM “1864”/AA

We were inspired by Wang system [31], who proposed to use a BLSTM-RNN for a tagging solution that can be applied to various tagging tasks. Given a sentence $w_1, w_2, ..., w_n$ with labels $y_1, y_2, ..., y_n$, the BLSTM must predict the probability distribution of the labels for each word. Then, a decoder makes it possible to produce the labels of each input sequence. At each time step, the BLSTM network takes a vector of characteristics of the current word w_i. It is a binary vector of dimension $|V|$ where V is the vocabulary. The BLSTM layer integrates background and future information when predicting the current word and is updated according to the entire input phrase. The output layer is a Softmax layer whose size is the number of labels. It produces the tag probability distribution of the input word w_i as $y_1^{'}, y_2^{'}, ..., y_n^{'}$. All weights are trained using the back-propagation algorithm, to maximize the probability of the learning data. Since the music documents are composed of a set of repeated words, the vocabulary is small. If some words are outside the vocabulary, they are coded as “unknown”, in order to not disturb the tagging.

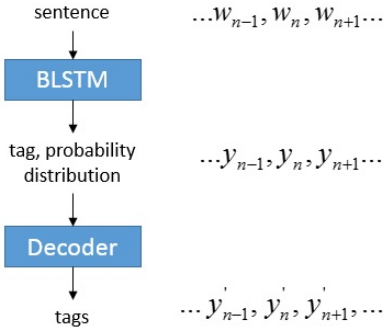


Fig. 4: Tagging system, from [31].

The tagger architecture consists of a single hidden layer with 50 hidden states. We have fixed the number of words to be retained in the vocabulary at 5000 and the number of preceding and following words to be used for the prediction, at 3. The network is driven with a back-propagation algorithm and evaluated at every 100 time steps. For the decoding part, given a phrase as input, the network chooses for each word, the most likely label.

The LSTM block operates in 4 steps which we will detail below:

In the first step, it decides which information to eliminate from the cell state (door of oblivion). It looks at h_{t-1} and x_t, and produces a value between 0 and 1 for each number

in the cell state C_{t-1}. If 1, it keeps this completely, and if 0, it gets rid of it completely.

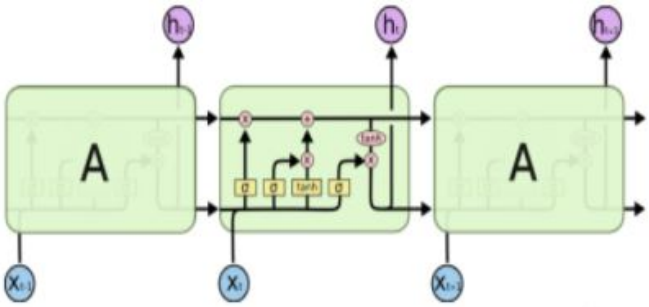
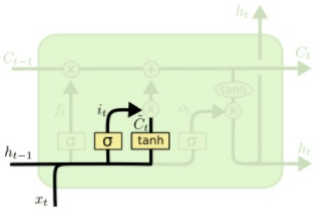


Fig. 5: Sequence of LSTM blocks, from [24].

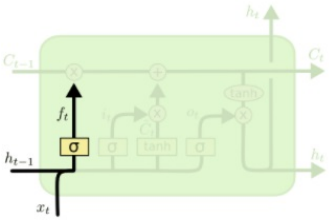
For example, let's try to predict the next word based on all the previous ones. In such a case, the cell state may include the current subject's gender so that correct pronouns can be used. When we see a new subject, we would like to forget the gender of the old subject. The formula corresponds to the door of forgetfulness (see Figure 6).



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Fig. 6: Formula of forgetfulness door [24].

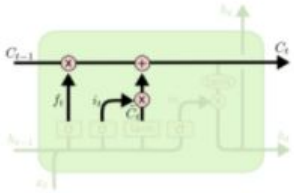
In the second step, it is a question of determining the new information to store in the cell state. This consists of two parts: 1) a Sigmoid layer called "gateway layer" that decides which values to update, and 2) a Tanh layer that creates a vector of new candidate values. For example, if we want to add the genus of the new subject to the cellular state, and to replace the old one that we forgot. The formula of the entrance gate layer and the Tanh layer is presented by the Figure 7.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Fig. 7: Formula of input door, from [24].

In the third step, we update the old cell state C_{t-1} in the new cell state C_t. We multiply the old state by f_t forgetting the things we decided to forget and then we add $i_t * C_t$ (entry gate * new candidate value). The new memory cell value is shown in Figure 8 [24].

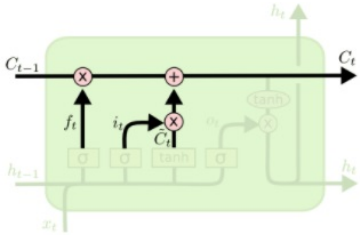


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Fig. 8: Formula of the new cellular state, from [24].

In the fourth step, this output will be based on our cell state, but will be a filtered version. We use a Sigmoid layer that determines which parts of the cell state we are going to produce. We put the cell state through Tanh (to push the values between -1 and 1) and multiply it by the exit of the Sigmoid gate, so that we produce only the parts we have decided.

For the example of the language model, since it has just seen a subject, it might want to display relevant information to a verb, if this happens next. For example, it can indicate whether the subject is singular or plural, so we know in what form a verb should be conjugated if follows next. The Figure 9 shows the output gate formula and the new value of the LSTM block.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Fig. 9: Output gate formula and the new LSTM block value, from [24].

For the decoder, after standardization of the BLSTM outputs by a Softmax function of CTC neurons, a path simplification operator is applied. This deletes the consecutive identical letters and also deletes the blank character where the decision is absent. Finally, an alignment of the obtained sequence can be carried out either by using a conventional dynamic programming algorithm or by developing a decoding via integrating a sentence model in the form of a statistical language model (n-gram).

4. Experiments

The evaluation of the system performance was carried out on two datasets: the workshop of violin and IAM [23]. The former comprises 651 lines, 6571 words including 827

proper nouns. The latter contains 300 lines, 1831 words including 120 proper nouns. For learning, a set of multi-writer handwriting lines were chosen, subject to line inclinations having 25% of lines for the test portion. Table 1 shows the accuracy rate for the extraction of proper nouns in the two datasets by distinguishing three classes. In class C1, we labelled only the proper nouns and the rest of the words are arranged in the same unknown class. In class C2, we labelled the proper names and some words with a local vocabulary, specific to the application. Finally, in class C3, we labelled all the words using a TreeTagger. This labelling is done to show the interest of the tagging, and then the influence of a specific tagging compared to a general one.

Datasets	C1: NP + None	C2: NP + Local Tags	C3: NP + TreeTagger
Music Museum #NP=827 #words=6571	96,15%	98,38%	98,78%
IAM #NP=120 #words=1831	95,83%	100%	% 95,83%

Table 1: NP Rate accuracy for each set used in the experiment

We tested the system on only a small set of data to see how it works. For the tagging system, we took the dataset with a quit noise and a uniform distribution to take into account the errors of the recognition. First, we counted the number of occurrences of each letter present in the test document and we put all the characters in a list. For example, if a document contains 3 times the letter a, 2 times the letter b and 5 times the letter c. This list will be as follows: “abaacbcccc” and it is very likely to choose the letter c as letter a or b. Then, for each word, the characters to be modified are taken randomly according to the uniform law. Then, for each character, one arbitrarily selects from the list of characters of occurrences a character to replace the other character. Figures 10 and 11 show the precision curves obtained.

During the training phase, it is important to monitor the convergence of the system in order to avoid over fitting. We note that the precision increases with time until reaching 100% for the learning phase and 91% for the test phase. It can also be seen that in the course of learning, the average error which is linked to the overall performance of the system, decreases to reach zero, after a few steps of time. It can be concluded that error is a good criterion for quality control of learning.

The overall performance of the system can be improved by adding a lexicon or other high-level information. Here, we have chosen not to allow any correction by simply applying an exact marking method. We also validate our BLSTM by

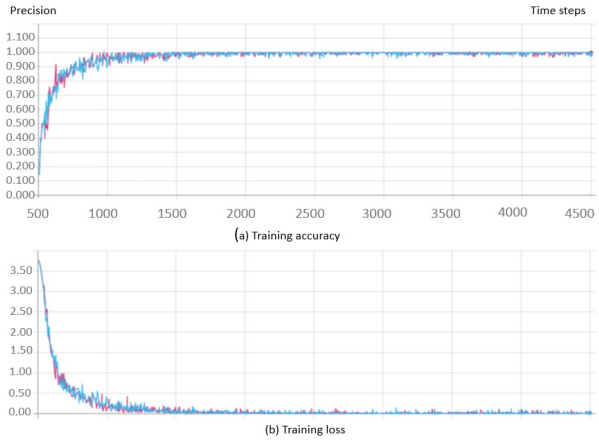


Fig. 10: Performances of tagging system on the violin workshop (blue color) and on IAM dataset (red color) during the training phase.

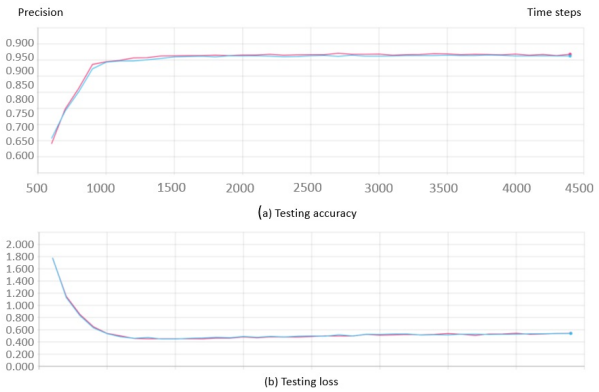


Fig. 11: Performances of tagging system on the violin workshop (blue color) and on IAM dataset (red color) during the testing phase.

performing line text queries. Among these queries we have, for example: “Vendu à Mr Angel à Elbeuf” and the result is “Vendu/VD à/PRP Mr/FP Angel/NP à/PRP Elbeuf/NV” with VD as sales class, PRP for proposal , FP for politeness, NP for proper noun and NV for city name.

Another example with noisy data: “Vanru à Me Maein à tolon” and one obtains “Vanru/<UNKNOWN> à/PRP Me/FP Maein/NP à/PRP tolon/<UNKNOWN>” where UNKNOWN indicates an unknown class. We observe, even if the data is noisy, that our system is able to find the proper noun. After several tests, we noticed that there are some contexts that have been well learned. Among these contexts, we find the formulas of politeness like “Mr”, “Mrs.”, “Miss”, etc., which lie before a proper noun. Also, if we have a preposition after a proper noun, then a city name will be found.

4.1 Implementation

For programming, we used Python language and the Tensorflow library. The nodes of the graph represent mathematical operations, while the edges of the graph represent the multidimensional data tables (tensors) communicated to each other. The flexible architecture allows us to deploy the calculation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. For the visualization of curves, we used TensorBoard, a tool linked to the Tensorflow library.

For the recognition system, the code is divided into four parts: A first part allows the extraction of image characteristics. A second part contains the architecture of the network with the set of parameters. A third part uses the network architecture and the characteristics extractor for driving the network and a last part for decoding. This means that, given an image, the network must be able to extract the characteristics and use the parameters learned to produce the text corresponding to the input image.

For tagging, the code is also divided into four parts: The first part is for processing the set of sentences by transforming the set of words into identifiers to facilitate data processing, then transforming each sentence into a vocabulary size vector (number of words), where for the word identifier indices if present in the sentence, one sets the value to 1 and the others to 0. For example, we have a vocabulary that contains the following words: this, is, cat, a and a sentence “this is a cat”.

```
- For "this", the vector is: [1,0,0,0]
- For "is", the vector is: [0,1,0,0]
- For  "a", the vector is: [0,0,0,1]
- For "cat", the vector is: [0,0,1,0]
```

The second part contains the architecture of the network with the set of parameters used. The third part uses the architecture of the network and the data processing part to train and evaluate the network.

5. Conclusion and perspectives

We have shown in this work how we can detect keywords by learning their left and right contexts, whatever the word. We avoided using word morphology and rigid syntax. The context can be free and even erratic. The part concerning the tagging of the words is original. It allows the second BLSTM to play its tagging role while recognizing the desired word. In the future, we would like to add more annotated data in order to better study the behavior of the system.

ACKNOWLEDGEMENTS

We are very grateful to the Museum of the Philharmonie de Paris for the sale transactions dataset that they provided at our disposal to realize this work.

References

[1] C. Adak, B. B. Chaudhuri and M. Blumenstein, Named Entity Recognition from Unstructured Handwritten Document Images, 2016, DAS, pp. 375-380.

[2] J. Almazan, A. Gordo, A. Fornes and E. Valveny (2014). Word spotting and recognition with embedded attributes. *TPAMI*, pp. 2552-2566.

[3] Y. Bengio, P. S. and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *Neural Networks*, 5(2):157-166.

[4] Brill, E. 1992. A Simple Rule-Based Part of Speech Tagger. In Proceedings of the DARPA Speech and Natural Language Workshop. pp. 112-116. Morgan Kauffman. San Mateo, California.

[5] H. Bunke (2003). Recognition of cursive roman handwriting - past present and future. In *7th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 448-459.

[6] G. Bideault, L. Mioulet, C. Chatelain and T. Paquet. Spotting Handwritten Words and REGEX using a two stage BLSTM-HMM architecture. Document Recognition and Retrieval, 2015, San Fransisco, USA.

[7] H. Daher, M. R. Bouguelia, A. Belaid, and V. Poulain d’Andecy, Multipage Administrative Document Stream Segmentation, ICPR, Stockholm, Sweeden, 2014, pp. 966-971.

[8] P. Doetsch, M. Kozielski and H. Ney. Fast and robust training of recurrent neural networks for offline handwriting recognition. In *ICFHR*. pp. 279–284. 2014

[9] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word Spotting and Recognition with Embedded Attributes. TPAMI, page 2552-2566, 2014.

[10] A. Fornes, V. RomeroâĂă, A. Baro, J. I. Toledo, J. A. Sanchez, E. Vidal and J. Llados, ICDAR2017 Competition on Information Extraction in Historical Handwritten Records, pp. 1389–1394.

[11] V. Frinkin, A. Fischer, M. Baumgartner and H. Bunke (2014). Keyword spotting for self-training of blstm nn based handwriting recognition systems. *Pattern Recognition*, 47:1073–1082.

[12] A. Graves, S. Fernandez, F. Gomez and J. Schmidhuber (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. *ICML*, pages 369-376.

[13] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke and J. Schmidhuber (2009). A novel connecsionist system for unconstrained handwritting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(5):855–868.

[14] S. España Boquera, M. J. Castro-Bleda, J. Gorbe-Moya and F. Zamora-Martinez (2011). Improving off-line handwritten text recognition with hybrid hmm/ann models. *IEEE TPAMI*, 33(4):767–779.

[15] J. Goodman (âĂŐ2001). A bit of progress in language modeling. *Computation and Language (cs.CL)*.

[16] A. Graves and J. Schmidhuber (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 18(5):602-610.

[17] S. Hochreiter and J. Schmidhuber (1997). Long short-term memory. *Neural computation*, 9(8):1735-1780.

[18] N. Kaji, Y. Fujiwara, N. Yoshinaga and M. Kitsuregawa (2010). Efficient staggered decoding for se-quence labelling nobuhiro kaji yasuhiro fujiwara naoki yoshinaga masaru kitsuregawa. In *Meeting of the Association for Computational Linguistics, Uppsa-la, Sweden, 11-16 July*, pages 485-494.

[19] N. Kooli and A. Belaïd (2016). Inexact graph matching for entity recognition in ocred docu-ments. In *ICPR*, pages 4071–4076.

[20] G. Lample, M. Ballesteros, S. Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. CoRR, abs/1603.01360, 2016.

[21] Z. C. Lipton and J. Berkowitz (2015). A critical review of recurrent neural networks for sequence learning. In *arXiv:1506.00019v4 [cs.LG]*.

[22] R. Manmatha and W. B. Croft (1997). Word spotting: Indexing handwritten manuscripts. *Intelligent Multimedia Information Retrieval, Mark T. Maybury, Ed. MIT Press, Cambridge, MA*, pages 43-64.

[23] U.-V. Marti and H. Bunke (2002). The IAM-database: an english sentence database for offline handwriting recognition. *IJDAR*, 5(1):39-46.

[24] Ch. Olah, (2015). Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[25] S. J. Pan and Q. Yang (2010). A survey on transfer learning. *IEEE Transactions on Knowledge Data Engineering*, 22(10):1345-1359.

[26] T. Plotzand and G. A.Fink (2009). Markov models for offline handwriting recognition: a survey. *International Journal on Document Analysis and Recognition (IJDAR)*, 12:269-298.

[27] T. M Rath and R. Manmatha (2007). Word spotting for historical documents. *IJDAR*, 9(2-4):139-152.

[28] B. Stuner, C. Chatelain, and T. Paquet, Self-Training of BLSTM with Lexicon Verification for Handwriting Recognition. ICDAR 2017, pp.633-638

[29] S. Sudholt and G. A. Fink (2016). Phocnet: A deep convolutional neural network for word spotting in handwritten documents. In *International Conference on Frontiers on Handwriting Recognition*, pages 277-282.

[30] K. Vesely, A. Ghoshal, L. Burget and D. Povey (2013). Sequence-discriminative training of deep neural networks. In *INTERSPEECH*, pages 2345-2349.

[31] P. Wang, Y. Qian, F. K. Soong, L. He and H. Zhao (2015). Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *ACL*.